TCS IPA PREPARATION 2025

UNIX MCQs

1. The command used to find the count of only words in a file?

a. wc –w <filename>

b. wc –words <filename>

c. wc <filename>

d. None of the above

Correct Answer: a

2. What will be the output of given awk script? awk 'BEGIN{s=0 while (s < 55) {prints;++s}}'

a. Prints the numbers from 0 to 55.

b. Prints the numbers from 0 to 5d.

c. Command will throw a syntax error

Correct Answer: C

3. What is the command to view content of file?

a. cat

b. ls

c. vi

Correct Answer: a

4. What is command to switch to another user?

a. su

b. switch

c. user

d. login

Correct Answer: a

5. What is command to set an environment variable?

a. env

b. export

c. Set

d. Unset

Correct Answer: b

6. What is command to change root directory?

a. chroot

b. chmod

c. chown

d. chgrp

Correct Answer: a

7. What is command to run process in background?

a. &

b. /

c. Ctrl + c

d. Ctrl + v

Correct Answer: a

8. Which Symbol be used with grep command to match the the pattern pat at the beginning of a line?

a . ^pat

b. $pat

c. pat$

d. pat^

Correct Answer: a

9. Which command is used to sort the lines of a data in a file in reverse order?

a sort

b. sh

c. sort -r

d. st

Correct Answer: c

10. Which of the command to change the protection mode of files starting with string emp and ending with 1,2 or 3?

a. chmod u+x emp[1-3]

b. chmod 777 emp

c. chmod u+r??? emp

d. chmod 222 emp?

Correct answer: a

11. Which command is used to remove the directory?

a. rd

b. rmdir

c. dl dir

d. rdir

Correct answer: b

12. Find the occurrences of word and print its occurrence count.

Answer: grep –o –i "unix" myfile | wc -l

13. What will be the output of the below command?

find –type f –perm 0777 -print

a. Display all files who has all the three permission for all levels of users

b. Display all directories who has all the three permission for all levels of users

c. Display all files who has minimum level of permission for all levels of users

d. Display all folders who has all the three permission for all levels of users

Correct answer: a

14. Which awk script will give you the average of second column of the file named file1, which has '|' as the delimiter?

a. awk "BEGIN(FS="";s=0)(s=s+$2) END(print s/NR)' file1

b. awk "BEGIN{FS="";s=0}{s=s+$2}END{print s/NR} file

c. awk 'BEGIN{FS="|";s=0}{s=s+$2}END{print s}' file1

d. awk "BEGIN{FS="";s=0}{s=s+$2}END{print s/NR} file1

Correct answer: c

15. Select the correct command to search "Hello Unix" in a file ignoring case?

a. grep "Hello Unix" file name

b. grep -i "Hello Unix" file name

c. grep -c "Hello Unix" file name

d. grep –i file name "Hello Unix"

Correct answer:b

16. Which of the following command in unix used to remove the duplicate lines?

a. sort -u

b. sort uniq

c. uniq

d. uniqe

Correct answer:a

Some of the most used commands in unix/Linux:

File & Directory Management

Command	Description
`ls`	Lists files and directories in the current location.
`cd`	Changes the current working directory.
`pwd`	Prints the current working directory path.
`mkdir`	Creates a new directory.
`rmdir`	Removes an empty directory.
`rm`	Deletes files or directories (use `-r` for directories).
`cp`	Copies files or directories.
`mv`	Moves or renames files or directories.
`touch`	Creates a new empty file or updates timestamp.
`find`	Searches files and directories based on conditions.
`locate`	Quickly finds files using an indexed database.

File Viewing & Editing

Command	Description
`cat`	Displays file content.
`more` / `less`	Views file content page-by-page.
`head`	Shows the first few lines of a file.
`tail`	Shows the last few lines of a file.
`nano` / `vi` / `vim`	Terminal-based text editors.

Compression & Archiving

Command	Description
`tar`	Archives files into a single .tar file.
`gzip` / `gunzip`	Compresses or decompresses `.gz` files.
`zip` / `unzip`	Compresses or extracts `.zip` files.

System Monitoring & Management

Command	Description
`top`	Displays running processes and system usage.
`htop`	Enhanced version of `top` (if installed).
`df -h`	Shows disk space usage.
`du -h`	Shows directory/file size.
`free -h`	Displays memory usage.
`ps`	Lists running processes.
`kill` / `killall`	Stops a running process.

Networking

Command	Description
`ping`	Checks network connectivity to a host.
`ifconfig` / `ip addr`	Displays network interfaces and IP addresses.
`netstat` / `ss`	Shows open ports and connections.
`curl`	Transfers data from or to a server (e.g., download files).
`wget`	Downloads files from the internet.

Package Management (Debian/Ubuntu)

Command	Description
`apt update`	Updates the package index.
`apt upgrade`	Installs available updates.
`apt install <package>`	Installs a package.
`apt remove <package>`	Removes a package.

User & Permission Management

Command	Description
`adduser` / `useradd`	Adds a new user.
`passwd`	Changes a user's password.
`whoami`	Shows the current logged-in user.
`sudo`	Executes a command with superuser privileges.

Miscellaneous

Command	Description
`echo`	Prints text to the terminal.
`date`	Displays or sets the system date and time.
`history`	Shows command history.
`alias`	Creates a shortcut for a command.
`clear`	Clears the terminal screen.

File Permissions & Ownership

Command	Description
`chmod`	Changes file permissions.
`chown`	Changes file ownership.
`umask`	Sets default permission for new files.

Symbolic File Permissions

	Read (4)	Write (2)	Execute (1)	Result
User	r	w	x	7
Group	r	-	x	5
Others	r	-	-	4

	Read (r)	Write (w)	Execute (x)	Result
User (u)	+	+	+	u+rwx
Group (g)	=	-	-	g=r
Others (o)	-	-	-	o-rwx
All (a)				

+ adds permission; - removes permission

= adds permission but removes other permissions;

Octal File Permissions

Octal Value	Mode	Octal Value	Mode
0	- - -	4	r - -
1	- - x	5	r - x
2	- w -	6	r w -
3	- w x	7	r w x

Octal Value	Symbolic Value	Result
777	a+rwx	rwxrwxrwx
755	u+rwx,g=rx,o=rx	rwxr-xr-x
644	u=rw,g=r,o=r	rw-r--r--
700	u=rwx,g-rwx,o-rwx	rwx------

Example COMMANDS:

For searching text and patterns:

grep (get regular expressions)

ex:

grep 'the' filename

grep -n 'the' filename (adds line count and displays words with 'the')

grep -i 'the' filename (case insensitive)

grep -v 'the' filename (except)

grep -E '{abcd}' filename (display all occurences of {abcd})

grep -E '\w{6,}' filename (display words having >=6 letters)

pipes to connect commands(|):

ex: echo "hello, dinesh" | wc (displays no.of lines,words,chars)

view text files with cat, head, tail and less:

ex:

cat filename (displays content in file)

head filename (displays first 10 lines)

tail filename (displays last 10 lines)

cat -n filename (add line count)

cat filename | tail filename | cat -n

head -n5 filename (displays 5 lines)

tail -n5 filename (displays last 5 lines)

manipulate text with awk,sed and sort:

ex:

awk '{print $2 "\t" $1}' filename (displays 2nd col and 1st col)

awk '{print $2 "\t" $1}' filename | sort -n (displays 2nd col and 1st col sorted with numeric values)

sed 's/orange/blue/' filename (changes the string orange to blue in the file)

SORT

sort filename (sorts accord to first char of 1st col)

sort -k2 filename (sorts accord to first char of sec col)(key)

sort -k2 -n filename (sorts accord to numeric values of sec col)

sort -u filename (displays only unique lines)(removes duplicates)

rev

Print text in reverse sequence.

tac

Concatenate and print files in reverse.

tr

Translate or modify individual characters according to parameters.

Creating tar and zip archives:

ex:

tar cvf filename.tar filestobearchived (makes a tar file)

tar caf filename.tar.gz filestobearchived (makes a compressed Zip)

tar caf filename.tar.bz2 filestobearchived (makes a compressed zip)

To unpack the tar file:

tar xf filename.tar.gz/bz2 -C foldername (unpacks into folder)

To make zip files:

zip filename.zip contentfilename

unzip filname.zip

UNIX Study Notes

1. Introduction to Unix

• Unix is a multi-user, multitasking operating system

• Developed in the 1970s at Bell Labs

• Known for its portability, stability, and security

• Uses a modular design with tools and utilities

• Basis for Linux, macOS, and other modern OS

Explanation:

Unix is a powerful, stable, and secure operating system originally developed for academic and industrial use. Its core strengths lie in multitasking and multi-user capabilities, allowing many users to operate and run processes simultaneously without conflict. It was designed with simplicity, efficiency, and the philosophy of building small programs that work together. Unix forms the foundation of many modern operating systems. Tools such as piping, redirection, shell scripting, and powerful command-line utilities make it ideal for automation,

server administration, and software development. Today, its influence extends through Linux distributions, BSD, and macOS.

Examples:

1. uname -a prints system information
2. Unix systems include Solaris, AIX, and Linux

2. File System Hierarchy

• Tree-structured with / as the root directory

• Key directories: /bin, /etc, /home, /usr, /var

• Device files are found in /dev

• Hidden files start with . and store configs

• Everything in Unix is treated as a file

Explanation:

The Unix file system is structured as a tree, beginning with the root directory /. All system components and user data are organized under this structure. Each directory has a specific purpose, such as /bin for essential binaries, /etc for configuration files, and /home for user-specific directories.

Understanding the hierarchy is crucial for navigation, system management, and scripting.

Treating everything—including hardware—as files brings consistency to input/output handling and allows for flexible operations using standard commands.

Examples:

1. /etc/hosts maps IP addresses to hostnames
2. /home/user/docs/report.txt is a user document

3. Basic File Navigation

• pwd shows current directory

• cd changes directories

• ls lists files and directories

• Use ls -l or ls -a for detailed/hidden views

• Tab completion improves speed

Explanation:

Navigation in Unix relies on basic commands. pwd displays the current directory path. cd moves between directories using absolute (/home/user) or relative (../) paths. ls is used to list files, while its options display more info or hidden files starting with ..

These commands are essential for accessing, organizing, or modifying files. Combined with keyboard shortcuts and autocompletion, they create a smooth and efficient terminal experience.

Examples:

1. cd /var/log moves to system logs

2. ls -l ~/Downloads shows contents of the Downloads folder

4. File Creation and Management

• touch creates files

• cp copies files

• mv renames or moves files

• rm deletes files/directories

• mkdir and rmdir manage folders

Explanation:

File management in Unix is done using a set of intuitive commands. touch is used to create

empty files. cp copies files or directories. mv is used for renaming or moving. rm deletes files and mkdir creates new folders. Options like -r (recursive) extend functionality.

It's important to be careful with commands like rm -rf, which can delete directories permanently without confirmation. File operations are fast, scriptable, and easily combinable with wildcards and filters.

Examples:

1. cp notes.txt backup/ copies to backup folder
2. rm -rf temp/ deletes the entire directory

5. File Viewing and Paging

• cat shows entire file content

• more and less display files one page at a time

• head shows the first 10 lines by default

• tail shows the last 10 lines

• tail -f follows live logs

Explanation:

When viewing large files, paging tools like less and more allow users to scroll through content without loading everything at once. head and tail provide quick access to the start or end of a file. This is particularly useful for log inspection.

These commands are also commonly used in pipelines with filters, allowing users to extract specific information from large files or stream real-time outputs during troubleshooting.

Examples:

1. less /var/log/syslog scrolls through logs
2. tail -f app.log displays real-time updates

6. File Permissions and Ownership

• Permissions: read (r), write (w), execute (x)

• Three groups: user, group, others

• chmod changes file permissions

• chown changes file owner

• umask sets default permissions

Explanation:

Unix files have permission sets for the owner (user), group, and others. Each can be allowed or denied read, write, and execute access. chmod changes these permissions numerically (e.g., 755) or symbolically (e.g., u+x). chown changes ownership.

Controlling permissions ensures data security and prevents unauthorized actions.

Understanding permission structures is essential for system security, collaborative environments, and proper automation.

Examples:

1. chmod 700 script.sh gives full permission to the user only

2. chown root config.txt assigns file to the root user

7. Filters and Search

• grep searches for patterns

• cut extracts fields from files

• sort and uniq organize text

• wc counts lines, words, or characters

• Filters can be piped for complex tasks

Explanation:

Text filters are essential for analyzing logs, configuration files, and output streams. grep is powerful for locating patterns using regular expressions. cut, sort, and uniq help organize and clean data. These commands are often combined via pipes to form efficient data pipelines.

Used together, these commands reduce the need for scripting or GUI-based tools when dealing with large datasets or automation workflows.

Examples:

1. grep "error" logfile.txt shows error entries

2. cut -d ',' -f1 users.csv | sort | uniq filters unique names

8. Redirection and Pipes

• > writes output to a file

• >> appends output

• < takes input from a file

• | passes output of one command to another

• Used extensively in scripting

Explanation:

Redirection and piping let Unix users build powerful command chains. You can redirect command output into a file or read from one using <. Pipes (|) pass output from one command directly into another, eliminating temporary files and increasing performance.

This concept is foundational to Unix's modularity. Redirection also helps automate workflows like saving logs, filtering output, and feeding commands into scripts.

Examples:

1. echo "Hello" > greet.txt saves text to a file

2. cat file.txt | grep "name" finds matching lines

9. User and Group Management

• whoami, id, groups check user info

• adduser, deluser manage users

• passwd sets passwords

• sudo grants elevated privileges

• usermod updates user info

Explanation:

Unix is built for multi-user environments. Tools exist for creating, modifying, and deleting users. Permissions and groups control access to system resources. sudo provides temporary administrative rights without giving full root access.

User management ensures secure access control and isolation between users. Proper group assignment allows for shared file access without compromising privacy.

Examples:

1. sudo adduser devuser creates a new user

2. groups devuser lists all groups the user belongs to

10. Shell and Shell Types

• The shell is a command-line interface to Unix

• Common shells: sh, bash, zsh, csh

• Shell interprets commands and scripts

• Each shell has unique syntax and features

• Bash is the most widely used shell

Explanation:

The shell in Unix is the interface between the user and the kernel. It interprets typed commands or scripts and executes them. While sh (Bourne shell) is the original, bash (Bourne Again Shell) became popular due to its features, scripting abilities, and widespread support.

Other shells like zsh or ksh offer enhanced scripting or interactive capabilities.

Understanding the shell's features allows developers to use command-line tools efficiently and automate tasks.

Examples:

1. echo $SHELL shows the current shell
2. bash script.sh runs a Bash script

11. Processes in Unix

• A process is a running instance of a program

• ps lists current processes

• top provides a live view of system activity

• kill terminates processes

• Each process has a PID (Process ID)

Explanation:

Processes represent running programs or scripts in Unix. The system tracks each process with a unique PID. Tools like ps and top are used to monitor or manage them. Foreground and background processes can be controlled using &, fg, and bg.

Managing processes is essential for performance, stability, and resource usage.

Understanding how to start, stop, or prioritize processes helps in troubleshooting and system

administration.

Examples:

1. ps aux | grep apache finds the Apache process

2. kill 1234 stops the process with PID 1234

12. Job Control and Background Tasks

• Use & to run tasks in background

• jobs lists current background jobs

• fg brings background job to foreground

• bg resumes paused job in background

• Ctrl+Z suspends current job

Explanation:

Unix supports multitasking even within a single terminal. Commands can be run in the background using &, allowing users to continue other work. Suspended jobs can be resumed or moved between foreground and background.

These capabilities are useful when executing long-running scripts or commands that don't require immediate attention. Job control improves productivity and system interactivity.

Examples:

1. ./longtask.sh & runs script in background

2. fg %1 resumes first background job

13. Shell Scripting Basics

• Shell scripts automate repetitive tasks

• Start with #!/bin/bash

• Use variables with $ and commands with $(...)

• Use if, while, for, and case for logic

• Scripts must be executable with chmod +x

Explanation:

Shell scripting uses shell commands in sequence to perform tasks automatically. Scripts can include control flow, logic, and text manipulation. They're often used for backups, monitoring, deployment, and configuration.

By writing reusable scripts, users can avoid repeating tasks, minimize errors, and ensure consistency in execution. Shell scripting is powerful yet simple for system-level automation.

Examples:

1. #!/bin/bash; echo "Welcome" prints greeting

2. if [$USER == "root"]; then echo "Admin"; fi

14. Conditional Statements in Shell

• if, else, and elif provide decision logic

• Conditions use [ ] or [[ ]]

• Use -eq, -ne, -lt for integers

• Use = and != for strings

• Nesting enables complex logic

Explanation:

Conditionals allow scripts to make decisions during execution. Numeric and string comparisons are common, and conditions can be nested or chained. Shell also supports logical operators (&&, ||) for combining conditions.

Conditionals enhance automation by tailoring script behavior based on input, environment, or system state. Proper use improves script flexibility and control.

Examples:

1. if [$age -gt 18]; then echo "Adult"; fi

2. if ["$status" = "ok"]; then echo "Success"; else echo "Fail"; fi

15. Looping in Shell Scripts

• for, while, and until loops are supported

• Useful for repeating commands over items or conditions

• break exits a loop early

• continue skips current iteration

• Loops can iterate over files, numbers, or arrays

Explanation:

Loops are used in shell scripts to repeat tasks efficiently. A for loop can iterate over a list of values, while a while loop runs based on a condition. Loops reduce code duplication and support batch processing or monitoring.

They’re essential for writing scripts that handle dynamic input or system events. With loops, scripts become more powerful and capable of handling large-scale tasks automatically.

Examples:

1. for i in 1 2 3; do echo $i; done prints numbers

2. while [$i -le 5]; do echo $i; i=$((i+1)); done